

Formal Management of CAD/CAM Processes

Michael Kohlhase

Johannes Lemburg
Ewaryst Schulz

Lutz Schröder

DFKI Bremen, Germany

FM 2009



- ▶ Analogies between engineering design process and software engineering processes exist **theoretically**.
- ▶ In practice, final ‘implementation’ step dominant
 - ▶ has well-developed tool support: **CAD systems**
- ▶ Idea: break this dominance by providing an integrated development methodology
 - ▶ formal, semi-formal, informal documents
 - ▶ tool support at all levels
 - ▶ **invasion** into CAD tools to provide user interfaces
- ▶ Context: **FormalSafe** project at DFKI
 - ▶ Comprehensive framework for document-oriented development process
- ▶ Today: experiments in **formal verification** of CAD objects

Why?



- ▶ Formal verification of physical properties
- ▶ Tracing of (formalized) requirements
- ▶ Improved control over the coherence of designs
- ▶ Semantically founded change management.

System of design stages (simplified from **VDI 2221**):

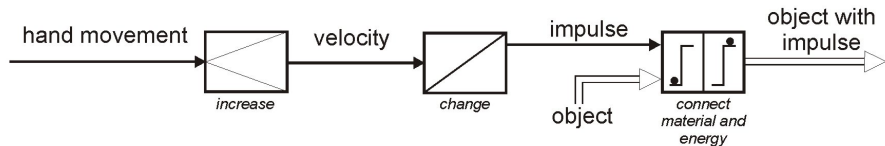
- ▶ Purpose
- ▶ Requirements
- ▶ Functional structure
- ▶ Solution in principle
- ▶ Embodiment Design

Current shortcomings:

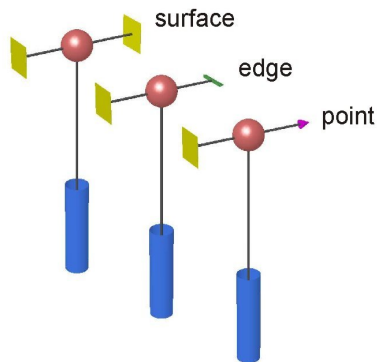
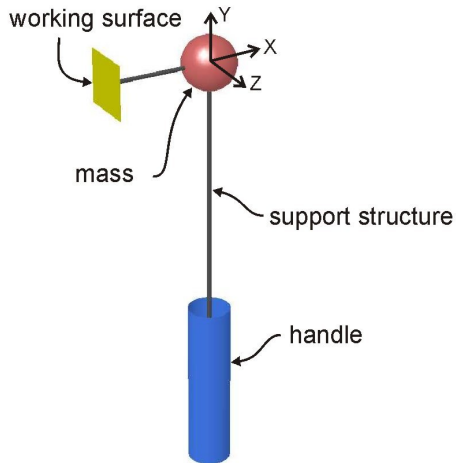
- ▶ No V-model style upwards feedback
- ▶ No verification of **correctness** of refinement steps
- ▶ Appreciable tool support only for last step: **CAD**

*A **hammer** is an apparatus for the manual generation and transmission of a defined impulse to an object, e.g. for driving a nail into a wall.*

The Function Structure of a Hammer



The Principle Solution for a Hammer



The Embodiment of a Hammer

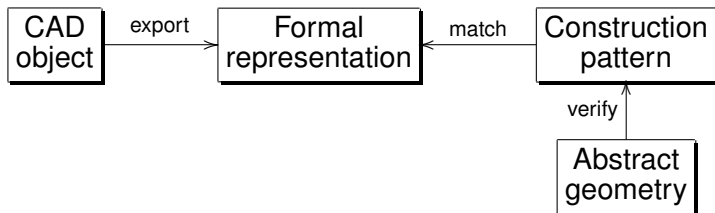


Sichere Kognitive Systeme



- ▶ **Invasive** approach:
Direct access to data structures of the CAD system
- ▶ Plug-in programmed using the SolidWorks API
- ▶ Presently: **export** of CAD objects into HASCASL.

- ▶ Regard CAD data structures as **syntax**, modelled as an algebraic datatype
- ▶ Provide a background modelling of abstract geometry, such as **affine real geometry**
- ▶ Define the **semantics** of CAD objects as point sets in affine space \mathbb{R}^3
- ▶ **Verify** properties of objects (using Isabelle/HOL within Hets), via intermediate **construction patterns**



spec SOLIDWORKS = AFFINEREALSPACE3DWITHSETS

then free types

Plane ::= Plane (o : Point; normal : VectorStar; innerCS : Vector);

Arc ::= Arc (center : Point; start : Point; end : Point);

Line ::= Line (from : Point; to : Point);

Spline ::= Spline (ps : List Point);

SketchObject ::= type Arc | type Line | type Spline;

Sketch ::= Sketch (objects : List SketchObject; pl : Plane);

Extrusion ::= Extrusion (sketch : Sketch; depth : Real);

...

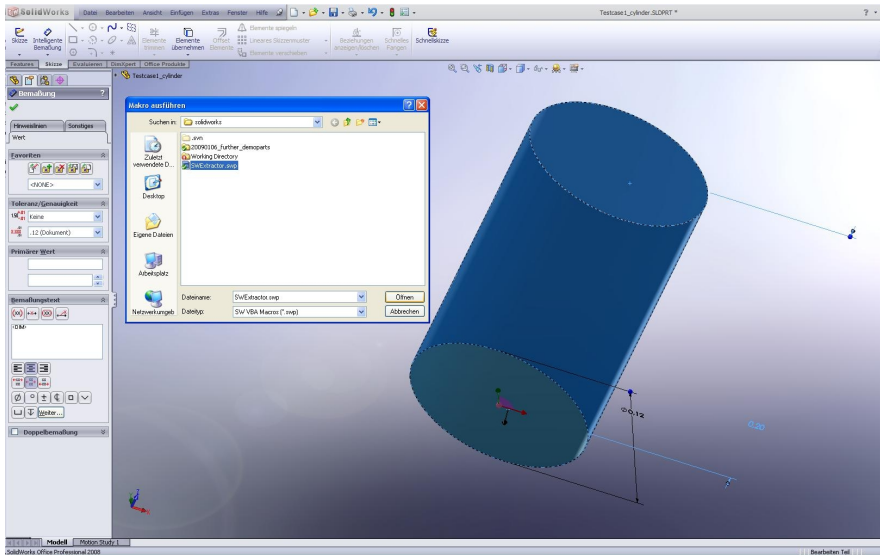
Feature ::= type Extrusion | ...

```
spec AFFINESPACE[VECTORSPACE[FIELD]] =  
  type   Point  
  op     --+-- : Point × Space → Point           %(point space map)%  
  vars   p, q : Point; v, w : Space  
  • p + v = p + w ⇒ v = w                       %(plus injective)%  
  • ∃ y : Space • p + y = q                       %(plus surjective)%  
  • p + (v + w) = p + v + w;                     %(point vector plus associative)%  
then %implies  
  ∀ p : Point; v, w : Space  
  • p + v + w = p + w + v;                       %(point vector plus commutative)%  
end
```

```
spec EXTAFFINESPACE [AFFINESPACE[VECTORSPACE[FIELD]]] = %def  
  op     vec : Point × Point → Space  
  ∀ p, q : Point • p + vec (p, q) = q;           %(vec def)%
```

ops $VWithLength(v : Vector; s : Real) : Vector =$
 v when $v = 0$ else $(s / (\| v \| \text{ as NonZero})) * v;$
 $Extrude(ax : Vector; ps : PointSet) : PointSet =$
 $\lambda x : Point \bullet \exists l : Real; y : Point$
 $\bullet l \in \text{closedinterval}(0, 1) \wedge y \in ps \wedge x = y + l * ax;$
 $i : Extrusion \rightarrow PointSet;$
 $i : Plane \rightarrow PointSet$
 $i : SketchObject \times Plane \rightarrow PointSet;$
 $i : Sketch \rightarrow PointSet;$
 \dots
 $\bullet i(Extrusion(sk, l))$
 $= Extrude(VWithLength(normal(Plane sk), l), i sk);$

Exporting a Cylinder: Before



The screenshot displays the SolidWorks interface. On the right, a blue cylinder is shown in a 3D perspective view. A dimension of $\varnothing 0.12$ is visible on the cylinder's surface. In the center-left, a dialog box titled "Makro ausführen" (Run Macro) is open. The "Suchen in" (Search in) field is set to "solidworks". The file list shows "SWExtractor.sld" selected. The "Dateiname:" (Filename) field contains "SWExtractor.sld" and the "Dateityp:" (File type) is "SW VBA Macro (*.mcp)". The "Dateiname:" field also has a dropdown menu showing "SWExtractor.sld" and "SWExtractor.sld". The "Dateityp:" field has a dropdown menu showing "SW VBA Macro (*.mcp)". The "Dateiname:" field also has a dropdown menu showing "SWExtractor.sld" and "SWExtractor.sld". The "Dateityp:" field also has a dropdown menu showing "SW VBA Macro (*.mcp)".

Slightly abstracting, have something matching the following **pattern**:

```
spec SOLIDWORKSCYLBYARCEXTRUSION = SOLIDWORKS
```

```
then op
```

```
SWCylinder(center, boundarypt : Point; axis : NZVector): Feature =  
Extrusion (Sketch ([ Arc (center, boundarypt, boundarypt) ],  
                Plane (center, axis, V (0, 0, 0))),  
            || axis ||)
```

spec CYLINDER = AFFINEREALSPACE3DWITHSETS

then op $Cylinder(base : Point; r : PReal; ax : NZVector) : PointSet =$
 $\lambda x : Point \bullet let v = vec (base, x) in$
 $\quad || proj (v, ax) || \leq || ax ||$
 $\quad \wedge || orthcomp (v, ax) || \leq r$
 $\quad \wedge (v * ax) \geq 0;$

view SWCYLBYAE_ISCYLINDER : CYLINDER **to**

{SOLIDWORKSCYLBYARCEXTRUSION

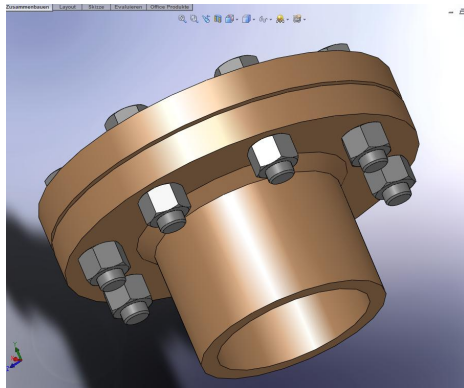
then op

$Cylinder(base : Point; r : PReal; axis : NZVector) : PointSet =$
 $let boundary = \lambda p : Point \bullet let v = vec (base, p)$
 $\quad in orth (v, axis) \wedge || v || = r;$
 $\quad boundarypt = choose boundary$
 $in i (SWCylinder (base, boundarypt, axis));$

}

- ▶ Export to HASCASL turns CAD objects into **fully formal documents**
 - ▶ Amenable to formal verification
 - ▶ Semantic change management
 - ▶ Rapid prototyping
- ▶ Proof of concept: formal verification of a cylinder
 - ▶ Several hundred lines of Isabelle/HOL

- ▶ Scalability
 - ▶ Make use of **modularity**
 - ▶ Library of **patterns**
- ▶ Methodology
 - ▶ Verification of **designs** against **solutions in principle**
 - ▶ **Specify** and verify **functional properties**
 - ▶ E.g. flanges according to norm DIN 1591-1



That's it.



Thanks for your attention!